A Heuristic for Generating Scenario Trees for Multistage Decision Problems

Kjetil Høyland * Michal Kaut † Stein W. Wallace ‡

June 20, 2000; revised April 6, 2001

Abstract

In stochastic programming models we always face the problem of how to represent the uncertainty. When dealing with multidimensional distributions, the problem of generating scenarios can itself be difficult. We present an algorithm for efficient generation of scenario trees for single or multi-stage problems.

The presented algorithm generates a discrete distribution specified by the first four marginal moments and correlations. The scenario tree is constructed by decomposing the multivariate problem into univariate ones, and using an iterative procedure that combines simulation, Cholesky decomposition and various transformations to achieve the correct correlations.

Our testing shows that the new algorithm is substantially faster than a benchmark algorithm. The speed-up increases with the size of the tree and is more than 100 in a case of 20 random variables and 1000 scenarios. This allows us to increase the number of random variables and still have a reasonable computing time.

Keywords: stochastic programming, scenario tree generation, asset allocation, Cholesky decomposition, heuristics

1 Introduction

Gjensidige Nor Asset Management (GNAM) has NOK 65 billion (7 billion US$) under management. During the last few years they have used stochastic-programming-based asset allocation models in their asset allocation processes. The most important step in that process is to establish the market expectations, i.e. to establish what are their beliefs for the major asset categories (bonds, stocks, cash, commodities and currencies) in different major regions of the world. The decision-makers prefer to express their expectations in terms of marginal distributions of the return / interest rates for the different asset classes in addition to correlations. The challenge in converting these expectations into an asset allocation mix is twofold: First we need to convert the expectations into a format which a stochastic programming model can handle, second we need an optimisation model which gives us the optimal asset mix, given this input.

Practical experience has told us that the first part, the scenario generation, can in fact be the most challenging and (computer-) time consuming one. For the purpose of generating the input data, GNAM has been using the method described in *Høyland and Wallace, 2001*, a method developed in 1996. For larger problems with many asset classes the scenario generation became the bottleneck in the portfolio optimisation process. This paper presents an algorithm that reduces the computing time for the scenario generation substantially.

The most well-known applications of asset allocation models are the Russell Yasuda Kasai model in *Cariño and Ziemba, 1998* and the models implemented by Towers Perrin in *Mulvey, 1996*. Other applications can be found in *Consigli and Dempster, 1998* and *Dert, 1995*.

These models are all focused on long term strategic asset liability planning. Stochastic processes are widely used for scenario generation in such models. The big challenge with such

*Gjensidige Nor Asset Management, POBox 276, N-1326 Lysaker, Norway

†Dept. of Industrial Economics and Technology Management, NTNU, N-7491 Trondheim, Norway

‡Centre for Advanced Study, Norwegian Academy of Science and Letters, N-0271 Oslo, Norway

processes is to calibrate the parameters so that the generated scenarios are consistent with the decision-maker's beliefs of the future development of the asset classes. In many applications the parameters are calibrated so that the future scenarios are consistent with the past. This might be acceptable for long term strategic planning. For tactical short term planning, i.e. for the question of how to construct an asset allocation mix relative to an asset allocation benchmark, such approaches are, however, inappropriate. The user wishes to express views on the future which deviate from the past. It is then important that the decision-maker can express the market expectations in a way that he or she finds convenient and that these expectations are converted to model input in a consistent manner.

This is the basic philosophy of the scenario generation method proposed in *Høyland and Wallace, 2001*. The user specifies his or her market expectations in terms of marginal distributions for each asset class in addition to correlations between the different asset classes and possibly other statistical properties. The stochastic, possibly multi-period, asset allocation model requires discrete outcomes for the uncertain variables. To generate these outcomes a regression model is applied. The idea is to minimise the distance between the statistical properties of the generated outcomes and the specified statistical properties (specified either directly or derived from the marginal distributions), see *Høyland and Wallace, 2001* for more details.

In the general form of the algorithm presented in *Høyland and Wallace, 2001* all the random variables (returns for all the asset classes) are generated simultaneously. Such an approach becomes slow when the number of random variables increases. In this paper we present an algorithm that decomposes the scenario generation problem. We generate one marginal distribution at a time and create the multivariate distribution by putting the univariate outcomes 'on top of each other', i.e. by creating the i-th multivariate outcome as a vector of the i-th outcomes of all the univariate distributions. We then apply various transformations in an iterative loop to reach the target moments and correlations.

The presented algorithm is inspired by the work of *Fleishman, 1978*, *Vale, 1983* and *Lurie and Goldberg, 1998*. Fleishman presents a cubic transformation that transforms a sample from a univariate normal distribution to a distribution satisfying some specified values for the first four moments. Vale and Maurelli address the multivariate case and analyse how the correlations are changed when the cubic transformation is applied. The algorithm assumes that we start out with multivariate normals. The initial correlations are adjusted so that the correlations after the cubic transformation are the desired ones. The algorithm is only approximate with no guarantee about the level of the error.

Lurie and Goldberg outlined an algorithm that is of similar type as ours. They also generate marginals independently and transform them in an iterative procedure. There are, however, two major differences between the two algorithms. One is in the way they handle the (inevitable) change of distribution during the transition to the multivariate distribution – while they modify the correlation matrix in order to end up with the right distribution, we modify the starting moments. The other major difference is that they start out with parametric univariate distributions whereas we start out with the marginal moments. We believe that specifying marginal moments is a more flexible approach and we certainly could also derive the marginal moments (up to the desired number) from the parametric distribution and apply our approach.

In Section 2 we present the algorithm. To simplify the presentation the algorithm is illustrated for the single period case. Høyland and Wallace argue that the quality[1] of the scenario tree improves if several small trees are aggregated to a larger tree rather than generating the larger tree directly. We present the algorithm for a single period with no aggregation. In Section 3 we discuss how the algorithm can be adjusted to incorporate tree aggregation and multiple periods. Numerical results are presented in Section 4, while possible future research areas are discussed in Section 5.

2 The algorithm

In Høyland and Wallace's method a scenario tree can in principle be constructed to fit any statistical properties. In this section we will assume that the specified properties are the first four marginal moments and the correlations. This is consistent with many other studies, as

[1] See *Høyland and Wallace, 2001* for how to measure 'quality' of the scenario tree.

well as our own empirical analyses in *Høyland and Wallace, 2001* of what were the important statistical properties. The presented methodology is more general than this, we could in principle specify even higher moments, but it is more restrictive than our original approach, where a wide range of statistical properties could be specified.

The general idea of the algorithm is as follows: Generate univariate outcomes for n random variables each satisfying a specification for the first four moments. Transform these outcomes so that the new outcomes are consistent with a correlation matrix. The transformation will distort the higher than the second moments of the original univariate outcomes. Hence we need to start out with a different set of higher moments so that the higher moments that we end up with are the right ones.

The procedure would lead to the exact desired values for the correlations and the marginal moments if the generated univariate outcomes were independent. This is, however, true only when the number of scenarios goes to infinity. With a limited number of scenarios, the marginal moments and the correlations will therefore not fully match the specifications. To be able to secure that the error is within a pre-specified range we have developed an iterative algorithm, which is an extension of the core algorithm.

Section 2.1 discusses the assumptions we have on the correlation matrix, Section 2.2 introduces necessary notation, Section 2.3 explains the key transformations used in the algorithm, Section 2.4 describes the core module of the algorithm, while Section 2.5 explains the iterative procedure.

2.1 Assumption on the correlation matrix

There are two assumption on the specified correlation matrix R. The first one is a general assumption that R is a possible correlation matrix, i.e. that it is a symmetric positive semi-definite matrix with 1's on the diagonal. While implementing the algorithm there is no need to write a special code that controls the positive semi-definiteness, because we do a Cholesky decomposition of the matrix R at the very start. If R is not positive semi-definite, the Cholesky decomposition will fail.

Note that having an R that is not positive semi-definite means having some internal inconsistency in the data, so we should re-run our analysis. As an alternative, there exist several algorithms that find a possible correlation matrix that is, in some sense, closest to the specified matrix R. One such an algorithm can be find in *Higham, 2000*. Another approach is used in *Lurie and Goldberg, 1998*, where a QP model is formulated to solve the problem. The latter approach has an advantage of being very flexible and allowing, for example, for specifying weights that express how much we believe in every single correlation. We can also used bounds to restrict the possible values. On the other hand, it has the obvious disadvantage of needing a QP solver.

The other assumption is that all the random variables are linearly independent, so R is non-singular – hence positive definite – matrix. For checking this property we can use again the Cholesky decomposition, because the resulting lower-triangular matrix L will have zero(s) on its diagonal in a case of linear dependency.

This is not a serious restriction, since having a linearly dependent variable mean that it can be computed from the others after the generation, and we can thus decrease the size of the problem. If we are not sure about our correlations, we can use the method from *Lurie and Goldberg, 1998* mentioned in the previous paragraph to enforce the positive definiteness. Since they have the elements of the Cholesky-decomposition-matrix L_{ij} as variables in the QP model, we can simply add a constraint $\{L_{ii} \geq \varepsilon > 0\}$ for every random variable i.

We also suggest a simple heuristic to handle the singularity of the correlation matrix inside our algorithm, see Section 2.5.1 It is, however, a very crude procedure, and we recommend to avoid singular correlation matrices if possible by going back to the analysis of the data.

2.2 Notation

To formulate the model we introduce the following notation. Note that vectors are columns by default.

n	number of random variables
s	number of scenarios
$\tilde{X}$	n-dimensional random variable

	$\rightarrow$ $\tilde{X}_i$ is an ith marginal of $\tilde{X}$
	$\rightarrow$ every moment of $\tilde{X}$ is a vector of size n
	$\rightarrow$ correlation matrix of $\tilde{X}$ is a matrix of size $n \times n$
X	one realisation of random variable $\tilde{X}$ – vector of size n
$\mathbb{X}$	matrix of s outcomes of random variable $\tilde{X}$ – $\mathbb{X}$ has dimension $n \times s$
$\mathbb{X}_i$	row vector of outcomes for random variable $\tilde{X}_i$ – $\mathbb{X}_i$ has size s
$\tilde{X}(mom; corr)$	$\tilde{X}$ has moments $mom = mom_1 \dots mom_4$ and a correlation matrix $corr$ – every mom_i is a vector of size n, and $corr$ is a matrix of size $n \times n$

Our goal is to generate outcomes $\mathbb{Z}$ from an n-dimensional random variable $\tilde{Z}$ with properties $\tilde{Z}(TARMOM; R)$, i.e. $TARMOM$ is the matrix (of size $4 \times n$) of target moments and R is the target correlation matrix (of size $n \times n$).

2.3 Key transformations

The core module, which will be presented in the next section, has two key transformations. One is a cubic transformation used for generating univariate distributions with specified moments. The other is a matrix transformation used to transform the multivariate distribution to obtain a given correlation matrix.

2.3.1 Cubic transformation

This transformation comes from *Fleishman, 1978*, where a method to simulate an univariate non-normal random variable $\tilde{Y}_i$ with given first four moments is introduced.[2] It takes a $\mathcal{N}(0,1)$ variable $\tilde{X}_i$ and uses a cubic transformation

$$\tilde{Y}_i = a + b\tilde{X}_i + c\tilde{X}_i^2 + d\tilde{X}_i^3$$

to obtain $\tilde{Y}_i$ with the target moments. Parameters a, b, c and d are found by solving a system of non-linear equations. Those equations utilise normality of the variable $\tilde{X}_i$.

The problem of this approach is that $\tilde{X}_i$ must have the first 12 moments equal to those of $\mathcal{N}(0,1)$ in order to get exactly the target moments of $\tilde{Y}_i$. Since this is hard to achieve, either by sampling or discretisation, the results with those formulas are only approximate. We have thus dropped the assumption of normality and derived formulas that work with arbitrary random variable $\tilde{X}_i$ – see Appendix C. Parameters a, b, c and d are now given by a system of four implicit equations.

We have used a simple mathematical-programming (regression) model to solve the system. In this model we have a, b, c and d as independent variables, moments of $\tilde{Y}_i$ as dependent variables, and we minimise a distance of the moments from their target values. Other possibilities for solving the system are for example using `Matlab` routines or some variant of Newton's method.

Note that unlike the normal case,[3] the parameters depend not only on the target moments of $\tilde{Y}_i$, but also on the first 12 moments of $\tilde{X}_i$. Our method for generating a sample $\mathbb{Y}_i$ from $\tilde{Y}_i$ is then following:

- take some sample $\mathbb{X}_i$ of the same size as $\mathbb{Y}_i$
- calculate the first 12 moments of $\mathbb{X}_i$ (as a discrete distribution)
- compute the parameters a, b, c, d
- construct $\mathbb{Y}_i$ as $\mathbb{Y}_i = a + b\mathbb{X}_i + c\mathbb{X}_i^2 + d\mathbb{X}_i^3$

2.3.2 Matrix transformation

Our other main tool in the algorithm is a matrix transformation

$$\tilde{Y} = L \times \tilde{X}$$

[2] The index i is obsolete in this section, but we use it for consistence with the rest of the paper.

[3] Parameters depend on the first 12 moments of $\tilde{X}_i$ also when $\tilde{X}_i$ is normal, but we know their values in advance.

where L is the lower-triangular matrix. The matrix L always comes from a Cholesky decomposition of the correlation matrix R, so we have $L \times L^T = R$.[4]

From the statistical theory we know that if $\tilde{X}$ is an n-dimensional $\mathcal{N}(0,1)$ random variable with correlation matrix I (and therefore with $\tilde{X}_i$ mutually independent), then the $\tilde{Y} = L \times \tilde{X}$ is an n-dimensional $\mathcal{N}(0,1)$ random variable with correlation matrix $R = L \times L^T$.

Since we do not have normal variables, we need a more general result. To make the formulas as simple as possible, we restrict ourselves to the case of zero means and variances equal to 1. In the beginning of Section 2.4 we show how to deal with this restriction. Note that $E[\tilde{X}] = 0$ leads to $mom_i = E[\tilde{X}^i]$. We will thus use the two interchangeably for the rest of the section.

In Appendix B we show that if the $\tilde{X}_i$ have zero mean, variance equal to 1, and are mutually independent, and if $R = L \times L^T$ is a correlation matrix, then $\tilde{Y} = L \times \tilde{X}$ is an n-dimensional random variable with zero means, variances equal to 1, and correlation matrix (which is also the variance-covariance matrix) R. Its marginals $\tilde{Y}_i$ have higher moments:

$$\begin{aligned} \mathbb{E}\left[\mathbb{Y}_i^3\right] &= \sum_{j=1}^{i} L_{ij}^3 \, \mathbb{E}\left[\mathbb{X}_j^3\right] \\ \mathbb{E}\left[\mathbb{Y}_i^4\right] - 3 &= \sum_{j=1}^{i} L_{ij}^4 \left(\mathbb{E}\left[\mathbb{X}_j^4\right] - 3\right) \end{aligned}$$

We will need also the opposite direction of the transformation:

$$\tilde{X} = L^{-1} \times \tilde{Y}$$

Since the above formulas are triangular, it is easy to invert them. The higher moments of $\tilde{X}$ can be then express as:

$$\begin{aligned} \mathbb{E}\left[\mathbb{X}_i^3\right] &= \frac{1}{L_{ii}^3} \left(\mathbb{E}\left[\mathbb{Y}_i^3\right] - \sum_{j=1}^{i-1} L_{ij}^3 \, \mathbb{E}\left[\mathbb{X}_j^3\right] \right) \\ \mathbb{E}\left[\mathbb{X}_i^4\right] - 3 &= \frac{1}{L_{ii}^4} \left[\mathbb{E}\left[\mathbb{Y}_i^4\right] - 3 - \sum_{j=1}^{i-1} L_{ij}^4 \left(\mathbb{E}\left[\mathbb{X}_j^4\right] - 3\right) \right] \end{aligned}$$

We divide only by the diagonal elements L_{ii} in these formulas. We can do it since L_{ii} are positive due to regularity (positive definiteness) of R. Note however that if L_{ii} are close to zero, we can experience numerical problems. See remark at the end of Appendix B for how to check for this problem in advance, or Section 2.5.1 for how to deal with it inside the algorithm.

Having both directions of the transformation allows us to transform random variables from one correlation matrix to another. First we transform the variables so that they are independent using the inverse formulas and then we use the original formulas to transform variables to the target correlation matrix. We will use that later in the algorithm.

2.4 The core algorithm

This section presents the core algorithm. It runs as follows: Find the target marginal moments. Generate n univariate discrete distributions that satisfy those moments. Create a multivariate distribution by putting the univariate outcomes 'on top of each other' in the order they coincidentally were generated. (We can do this since all the univariate distributions have the same size and the same fixed probabilities for the corresponding outcomes.) Transform these outcomes so that they have the desired correlations and marginal moments. If the discrete distributions $\mathbb{X}_i$ were independent, we would end up with a sample having exactly the desired properties.

For an easier orientation in the algorithm, we have divided it in two parts. In the *input phase* we read the target properties specified by the user and transform them to a form needed by the algorithm. In the *output phase* we generate the outcomes and transform them to the original properties.

[4] Note that L always exists, since we assume R to be positive semi-definite.

2.4.1 The input phase

In this phase we work only with the target moments and correlations, we do not have any outcomes yet. This mean that all the operations are fast and independent on the number of scenarios s.

Our goal is to generate an n-dimensional random variable $\tilde{Z}$ with moments $TARMOM$ and a correlation matrix R. Since the matrix transformation needs zero mean and variance equal to 1, we have to change the targets to match this demand. Instead of $\tilde{Z}$ we will thus generate random variables $\tilde{Y}$ with properties $\tilde{Y}(MOM; R)$, $MOM_1 = 0$, and $MOM_2 = 1$. $\tilde{Z}$ is then computed at the very end of the algorithm as

$$\tilde{Z} = \alpha\tilde{Y} + \beta$$

In Appendix A we show that the values leading to the correct $\tilde{Z}$ are:

$$\alpha = TARMOM_2^{1/2} \qquad MOM_3 = \frac{TARMOM_3}{\alpha^3}$$
$$\beta = TARMOM_1 \qquad MOM_4 = \frac{TARMOM_4}{\alpha^4}$$

The final step in the input phase is to derive moments of independent 1-dimensional variables $\tilde{X}_i$ such that $\tilde{Y} = L \times \tilde{X}$ will have the target moments and correlations. To do this we need to find the Cholesky-decomposition matrix L, i.e. a lower-triangular matrix L so that $R = L \times L^T$.

The input phase then contains the following steps:

1. Specify the target moments $TARMOM$ and target correlation matrix R of $\tilde{Z}$
2. Find the 'normalised moments MOM of $\tilde{Y}$
3. Compute L and find the transformed moments $TRSFMOM$ of $\tilde{X}$ — see Section 2.3.2

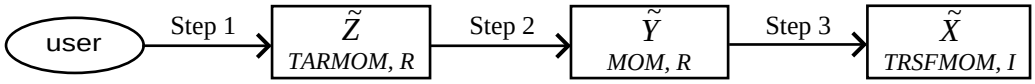

Figure 1: Input Phase

2.4.2 The output phase

In this phase we generate the independent outcomes, transform them to get the intermediate-target moments and the target correlations and finally transform them to the moments specified by the user. Since the last transformation is a linear one, it will not change the correlations. All the transformations in this phase are with the outcomes, so the computing time needed for this phase is longer and increases with the number of scenarios s.

We start by generating discretisations of the univariate distributions $\tilde{X}_i$ independently, one by one. This is a well-known problem and there are several possible ways to do it. We have used a method from *Fleishman, 1978*, in a way described in Section 2.3.1. The method starts by sampling from $\mathcal{N}(0, 1)$ and afterwards uses the cubic transformation to get the desired moments. For the starting $\mathcal{N}(0, 1)$ sample we use a random-number generator. An alternative would be to use a discretisation based on the distribution, or some other method for the starting $\mathcal{N}(0, 1)$ sample.

Another possibility is to use the method described in *Høyland and Wallace, 2001*. They formulate a non-linear regression model with outcomes of $\tilde{X}_i$ as independent variables and the first four moments as dependent variables. The distance (sum of squares) of the actual moments from the target moments is then minimised.

Once we have generated the outcomes $\mathbb{X}_i$ of marginals $\tilde{X}_i$, we can proceed with the transformations. First $\mathbb{Y} = L \times \mathbb{X}$ to get the target correlations and then $\mathbb{Z} = \alpha\mathbb{Y} + \beta$ to get the user-specified moments[5].

The output phase of the core algorithm consists of the following steps:

4. Generate outcomes $\mathbb{X}_i$ of 1-dimensional variables $\tilde{X}_i$ (independently for $i = 1 \dots n$)
5. Transform $\mathbb{X}$ to target correlations: $\mathbb{Y} = L \times \mathbb{X} \ \rightarrow \mathbb{Y}$ is $\tilde{Y}(MOM; R)$
6. Transform to original moments: $\mathbb{Z} = \alpha\mathbb{Y} + \beta \ \rightarrow \mathbb{Z}$ is $\tilde{Z}(TARMOM; R)$

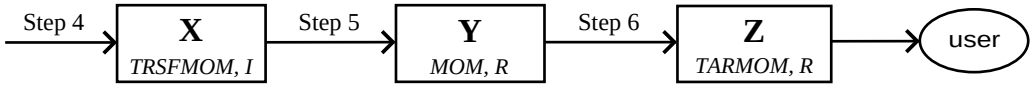

Figure 2: Output Phase

2.5 The modified algorithm

We know that the core algorithm gives us the exact results only if the random variables having support $\mathbb{X}_i$ are independent. Since we generate each set of outcomes $\mathbb{X}_i$ separately and have a limited number of outcomes (s), the sample co-moments will most certainly differ from zero and the results of the core algorithm will be only approximate.

The modified algorithm is iterative, so all results are approximate, with a pre-specified maximal error. We will again use the matrix transformation $\tilde{Y} = L \times \tilde{X}$, this time both forward and backward, as mentioned in 2.3.2. Recall that this transform allows us to obtain a desired correlation matrix, but it changes the moments while doing it.

In Section 2.3.1 we showed that the cubic transformation allows us to transform the general univariate random variable $\tilde{X}$ to a variable with some target moments. This transformation changes the correlations, but if the starting $\tilde{X}$ is close to the target distributions, the change in the correlations is expected to be small.

Our idea is thus to apply the cubic and the matrix transformation, alternately, hoping that the algorithm converges and outcomes with both right moments and correlations are achieved. This introduces the iterative loops to the core algorithm, namely to Steps 4 and 5, in the following way:

In Step 4 we would like to generate the independent outcomes $\mathbb{X}_i$. Since independence is very hard to achieve, we change our target to generating uncorrelated outcomes $\mathbb{X}_i$, i.e. we seek to get $\mathbb{X}$ with properties $\tilde{X}(TRSFMOM; I)$. We use an iterative approach to achieve those properties.

Since we do not control the higher co-moments, they will most likely not be zero, and the $\mathbb{Y}$ obtained in Step 5 will not match the target properties. We thus need another loop there to end up with the desired $\mathbb{Y}$.

The iterative versions of Steps 4 and 5 are[6]:

Step 4

4.*i.* Generate n marginals with target moments $TRSFMOM$ (independently)
→ we get $\mathbb{X}$ with correlation matrix R_1 close to I due to independent generation
4.*ii.* `let` $p = 1$ and $\mathbb{X}_1 = \mathbb{X}$
4.*iii.* `while` $dist(R_p; I) > \varepsilon_x$ `do`
4.*iv.* `do` Cholesky transformation: $R_p = L_p \times L_p^T$
4.*v.* `do` backward transform $\mathbb{X}_p^* = L^{-1} \times \mathbb{X}_p \ \rightarrow$ zero correlations, wrong moments

[5]Note that we have stopped to speak about distributions and started to speak about outcomes, so we have to change the notation from $\tilde{X}$ to $\mathbb{X}$

[6]As we have not found any better notation, in the rest of this section lower index p denotes an iteration counter, not a matrix column.

4.*vi.* `do` cubic transform of $\mathbb{X}_p^*$ with $TRSFMOM$ as the target moments;
store results as X_{p+1} $\rightarrow$ right moments, wrong corr.

4.*vii.* compute correlation matrix R_{p+1}

4.*viii.* `let` $p = p + 1$

4.*ix.* `let` $\mathbb{X} = \mathbb{X}_p$ $\rightarrow$ $\mathbb{X}$ is from $\tilde{X}(TRSFMOM; I)$ with correlation error $dist(R_p; I) \leq \varepsilon_x$

The sum of squares of differences of elements is used as a distance of two matrices in 4.*iii*. The same distance is used also at points 5.*iii* and 5.*ix* below. Since $\mathbb{X}$ is not a final output, the maximum error ε_x in Step 4 is typically higher than the corresponding ε_y in Step 5.

There are two possible outcomes from Step 4: X_p corresponding to random variables with right moments and wrong correlations, and X_{p-1}^* corresponding to random variables with slightly off moments and zero correlations. We start with the latter in the Step 5, and denote it X^*.

Step 5

5.*i.* $\mathbb{Y}_1 = L \times \mathbb{X}^*$ $\rightarrow$ wrong both moments and correlations
(due to higher co-moments different from zero)

5.*ii.* `let` $p = 1$ and let R_1 be the correlation matrix of $\tilde{Y}$

5.*iii.* `while` $dist(R_p; R) > \varepsilon_y$ `do`

5.*iv.* `do` Cholesky transformation: $R_p = L_p \times L_p^T$

5.*v.* `do` backward transform $\mathbb{Y}_p^* = L_p^{-1} \times \mathbb{Y}_p$
$\rightarrow$ zero correlations, wrong moments

5.*vi.* `do` forward transform $\mathbb{Y}_p^{**} = L \times \mathbb{Y}_p^*$
$\rightarrow$ right correlations (R), wrong moments

5.*vii.* `do` cubic transform of $\mathbb{Y}_p^{**}$ with MOM as the target moments;
store results as $\mathbb{Y}_{p+1} \rightarrow \mathbb{Y}_{p+1}$ has property $\tilde{Y}_{p+1}(MOM; R_{p+1})$

5.*viii.* `let` $p = p + 1$

5.*ix.* `let` $\mathbb{Y} = \mathbb{Y}_p$ $\rightarrow$ $\tilde{Y}$ has property $\tilde{Y}(MOM; R_p)$ with correlation error $dist(R_p; R) \leq \varepsilon_y$

Note that we can again choose from two different outcomes from Step 5 (and thus from the whole algorithm). After Step 5.*ix*, $\tilde{Y}$ has the right moments and (slightly) wrong correlation. If we prefer exact correlations, we can either use the last Y_p^{**}, or repeat Steps 5.*iv* – 5.*vi* just before we go to Step 5.*ix*.

Note also that Steps 5.*v* and 5.*vi* are written as individual steps for the sake of clarifying the presentation. Since we have always $s > n$ (usually $s \gg n$), it is much more efficient to join the two steps and do $\mathbb{Y}_p^{**} = \left(L \times L_p^{-1}\right) \times \mathbb{Y}_p$ directly.

2.5.1 Handling singular and 'almost-singular' correlation matrices

Since a singular R leads to zero(s) on a diagonal of the Cholesky-decomposition matrix L, we can not compute the inverse transformation in step 3 of the algorithm – see Section 2.3.2. In Appendix B we argue that in the case of an 'almost-singular' R we can evaluate the formulas, but the results will possibly be very inaccurate. We also suggest precautionary steps to avoid this problem. This section describes what to do when we nevertheless decide to proceed with the scenario generation.

The first concern if we expect an 'almost singular' correlation matrix is to implement the Cholesky decomposition in a stable way. *Higham, 1990* recommends an implementation with full pivoting and provides also a complete analysis of the stability including a bound for the backward-error.

The formulas found in Appendix B give us the best starting distribution $\tilde{X}$ for our iterative procedure to find $\tilde{Y}$. More precisely, a starting point, such that if we find independent $\tilde{X}$ with those moments, we will not need any iterations at all. Yet, it is not the only possible starting point. According to our tests, the convergence of the iterative procedure is very strong and the algorithm converges (almost) regardless the starting distribution $\tilde{X}$.

Therefore, if we are not able to find the starting $\tilde{X}_i$ (due to division by zero), or its calculated properties are extreme, or even impossible (for example a negative kurtosis), we can just replace it by some other starting distribution. The question of which starting distribution to choose depends on individual taste, and we offer two possibilities.

One is to substitute every 'wrong' number with a appropriate value from $\mathcal{N}(0,1)$, the other is to find the closest 'reasonable' value. For both of them we need to set up what we believe to be 'reasonable' values. Intervals like $E[\tilde{X}^3] \in (-3,3)$ and $E[\tilde{X}^4] \in (0.5,10)$ should be non-restrictive yet sufficient to avoid numerical difficulties.

The procedure for computing the target moments $TRSFMOM$ then is:

- Define the 'safe' intervals for $E[\tilde{X}^3]$ and $E[\tilde{X}^4]$
- `for i = 1 to n do`
- compute $E[\tilde{X}^3]$ and $E[\tilde{X}^4]$
- if $E[\tilde{X}^k]$ is out of range, replace it by
  a) the corresponding value from $\mathcal{N}(0,1)$
  b) the border value

3 Extensions

3.1 Tree aggregation

Høyland and Wallace, 2001 argue that the quality of the scenario tree improves if we generate several small trees and aggregate them to a large one, instead of generating the large tree directly. By aggregation they understand creating the large tree as the union of the scenarios of all the small trees.

It is straightforward to implement the tree aggregation in our algorithm. If we choose to generate u subtrees of size t (where $u\,t = s$), we simply call the generation procedure u times with t as the number of scenarios, collect the scenario sets from all runs and report the complete set of scenarios at the end. For guidance on the size of u and t see *Høyland and Wallace, 2001.*

3.2 Multi-period case

In the multi-period case there is much freedom as to how to generate a scenario tree. *Høyland and Wallace, 2001* recommend a sequential approach. They start out with the generation of the first period outcomes. The specifications for the second period will depend on the first period outcomes to capture empirically observed effects like mean reversion and volatility clumping. Hence the second period outcomes can be generated one at a time with specifications dependent on the first period outcomes. Third period outcomes are generated in a similar way, dependent on the path for the first and the second period outcomes. This process then continues until enough periods are generated.

The important observation here is that the multi-period tree is generated by constructing single period trees, one by one. In the same way, by generalising the single period case described in Section 2, the methods of this paper can address also the multi-period case.

4 Numerical results

This section presents the times needed to generate scenario trees with a varying number of random variables and scenarios. As a benchmark we use the algorithm from *Høyland and Wallace, 2001.* We have fixed the scenario probabilities in the benchmark algorithm to make the algorithms comparable.

Both algorithms are implemented in `AMPL` with `MINOS 5.5` as a solver. The test was done on a Pentium III 500 MHz machine with 256 MB memory, running Windows NT 4.0. As a distance used for evaluating the quality of the distribution in Steps 4 and 5 of the algorithm, we have used an absolute-mean-error defined as

$$MAE = \sqrt{\frac{1}{N_{el}} \sum_k (value_k - TARGET_k)^2}$$

where N_{el} is the number of elements in the sum. The distance was evaluated separately for moments ($N_{el} = 4n$) and for correlations ($N_{el} = \frac{n(n-1)}{2}$).

The stopping-values were $\varepsilon_x = 0.1$ ($\varepsilon_x = 0.2$ in the case of 20 r.v.) and $\varepsilon_y = 0.01$. Note that the distances are evaluated inside the algorithm, where all the data are scaled to variance equal to 1, so they are scale-independent. The formulas in Appendix A show that the maximum error of the i-th moment in the final output is $TARMOM_i^i \cdot \varepsilon_y$.

The following tables summarise the results. The two blank cells are left because the expected running time was very high.

	number of scenarios			
r.v.	40	100	200	1000
4	00:01	00:17	01:33	1:00:18
8	00:21	03:05	17:25	7:04:28
12	00:43	07:14	44:55	
20	03:47	37:22	4:00:00	

Results of the benchmark algorithm (`[h]:mm:ss`)

	number of scenarios			
r.v.	40	100	200	1000
4	00:01	00:01	00:01	00:05
8	00:04	00:04	00:03	00:10
12	00:09	00:07	00:06	00:16
20	01:05	00:44	00:31	00:48

Results of the new algorithm (`mm:ss`)

	number of scenarios			
r.v.	40	100	200	1000
4	1.1×	15×	67×	770×
8	5.8×	50×	354×	2605×
12	4.6×	60×	465×	
20	3.5×	51×	464×	

Speed-Up

We see that the new algorithm is always faster, with speed-ups ranging from almost nothing for the smallest tree to thousands for the biggest ones. The speed-up increases with both the number of scenarios and the number of random variables, even if there are exceptions to the latter case.

The most critical factor for the difference in running times is the number of scenarios. While the time for the benchmark model increases approximately quadratically[7], the time for our model increases less than linearly, in fact it decreases in most cases. This apparently strange behaviour is caused by a better convergence for larger trees. The number of iterations decreases with the size of a tree: with 40 scenarios we need typically 1–2 iterations in Step 4 of the algorithm (generation of $\tilde{X}$) and 2–3 iterations in Step 5 (generation of $\tilde{Y}$), whereas with 1000 scenarios we usually need only 1 iteration in both parts. Since we can not have less than 1 iteration, there will be no more improvement in the convergence for trees with more than 1000 scenarios. We can thus expect approximately linear time-dependency for those trees.

We should however realise that the above results are 'not fair' to the benchmark algorithm, because we do not use it the recommended way. Instead of trying to create a tree of 1000 scenarios directly, we should generate 10 trees of 100 (or 25 trees of 40) scenarios and aggregate them together, as described in Section 3.1, or in more detail in *Høyland and Wallace, 2001*. We should thus look at the speed-per-scenario. To make the results easier to read, the next table shows the fastest way to create scenario trees of different sizes. Note that it means to use the smallest tree (40 scenarios) for the benchmark model and the biggest tree (1000 scenarios) for our model.

	number of scenarios		
r.v.	benchmark	our alg.	speed-up
4	00:35	00:05	7.5×
8	08:39	00:10	53×
12	17:54	00:21	52×
20	1:34:46	00:51	113×

1000 scenarios with the best tree-aggregation (`[h]:mm:ss`)

[7] This is not surprising, since the number of specifications in the model is $4n + \frac{n(n-1)}{2} = 0.5(n^2 + 7n)$

5 Future work

Even though we have achieved a substantial increase of speed compared to the benchmark algorithm, there is still room for improvement. In the current `AMPL` implementation, the algorithm spends most of its time with the Cholesky transformation and the communication with the solver before and after the cubic transformation.

Both these critical times can be eliminated or decreased by implementing the algorithm in a compiled form (`C++`, `Matlab`, `Fortran`...) and without an external solver, instead of using an interpreted language like `AMPL`. Stable and efficient codes for Cholesky are widely available. The most difficult task is to find the coefficients of the cubic transformation. A `C++` implementation is currently being developed at the `SINTEF`[8], and according to the first tests it is more than 10 times faster than our `AMPL` code. Other approaches may be possible.

The presented algorithm is well suited for parallel implementation, because most of it can be processed independently for each random variable. The only step of the algorithm that uses considerable amount of time and can not be parallelised in this simple way is the Cholesky transformation of the correlation matrix, but parallel codes for the Cholesky transform can also be found.

Another area of future research is the scenario-tree aggregation. We will have to find out whether the recommendation to aggregate trees coming from *Høyland and Wallace, 2001* is valid also for our model. Note that while in their model there was a strong numerical argument for aggregating, our algorithm works better on larger trees. If we find it advantageous to aggregate, we will have to find the optimal number of trees to aggregate, given the number of random variables and scenarios. (See Section 3.1 for the details about how to aggregate the trees.)

6 Conclusion

We have presented an algorithm to generate scenario trees for multistage decision problems. The purpose of the algorithm is to speed up an existing scenario generation algorithm, which constructs multi-dimensional scenario trees that are consistent with specified moments and correlations.

The original algorithm constructs the multidimensional scenario tree by solving a single, potentially very large regression problem, based on least square minimisation. The main idea of the new algorithm is to decompose the regression problem so that each marginal distribution is constructed separately. To combine the different marginal distributions so that the multivariate distribution satisfies the specified correlations, we apply the Cholesky decomposition and a cubic transformation in an iterative procedure.

Testing shows that our algorithm is significantly faster, with speed-up increasing with the size of a tree. With 20 random variables and 1000 scenarios is our algorithm more than 100 times faster than the benchmark algorithm with no decomposition, assuming that we let the benchmark algorithm aggregate 25 trees of 40 scenarios instead of generating one tree of 1000 scenarios.

This improvement is of great practical importance since it allows for generating bigger trees and for more interactive use of decision models.

Acknowledgments

We would like to thank Erik Kole from the Maastricht University for pointing an error in the earlier version of the paper, and Matthias Nowak from `SINTEF`, Trondheim, for suggesting some important changes in notation and structure of the paper.

References

Cariño, D. R. and Ziemba, W. T. (1998). Formulation of the Russell-Yasuda Kasai financial planning model. *Operations Research*, 46(4):443–449.

[8]The Foundation for Scientific and Industrial Research in Trondheim, Norway

Consigli, G. and Dempster, M. A. H. (1998). Dynamic stochastic programming for asset-liability management. *Annals of Operations Research*, 81:131–162.

Dert, C. (1995). *Asset Liability Management for Pension Funds, A Multistage Chance Constrained Programming Approach*. PhD thesis, Erasmus University, Rotterdam, The Netherlands.

Fleishman, A. I. (1978). A method for simulating nonnormal distributions. *Psychometrika*, 43:521–532.

Higham, N. J. (1990). Analysis of the Cholesky decomposition of a semi-definite matrix. In Cox, M. G. and Hammarling, S. J., editors, *Reliable Numerical Computation*, pages 161–185, Walton Street, Oxford OX2 6DP, UK. Oxford University Press.

Higham, N. J. (2000). Computing the nearest correlation matrix. Numerical Analysis Report No. 369, Department of Mathematics, University of Manchester.

Høyland, K. and Wallace, S. W. (2001). Generating scenario trees for multi stage decision problems. *Management Science*, pages 295–307.

Lurie, P. M. and Goldberg, M. S. (1998). An approximate method for sampling correlated random variables from partially-specified distributions. *Management Science*, 44(2):203–218.

Mulvey, J. M. (1996). Generating scenarios for the Towers Perrin investment system. *Interfaces*, 26:1–13.

Vale, C. David & Maurelli, V. A. (1983). Simulating multivariate nonnormal distributions. *Psychometrika*, 48(3):465–471.

Appendix

A Linear transformation $\tilde{Z} = \alpha\tilde{Y} + \beta$

The purpose of the algorithm is to generate an n-dimensional random variable $\tilde{Z}$ with the target moments $TARMOM$. The algorithm needs the first two moments of all marginal $\tilde{Z}_i$ equal to 0 and 1 respectively, hence we generate the random variable $\tilde{Y}$ with $E[\tilde{Y}_i] = 0$ and $Var[\tilde{Y}_i] = 1$ instead, and compute $\tilde{Z}$ as $\tilde{Z} = \alpha\tilde{Y} + \beta$ afterwards.

For every marginal distribution $\tilde{Z}_i$ we need to find parameters α_i, β_i and moments MOM_{i3} and MOM_{i4} of $\tilde{Y}_i$, so that $\tilde{Z}_i = \alpha_i\tilde{Y}_i + \beta_i$ will have the desired moments $TARMOM_i$. We state the moments of $\tilde{Z}_i$ as functions of α_i, β_i and moments of $\tilde{Y}_i$, utilising $E[\tilde{Y}_i] = 0$ and $Var[\tilde{Y}_i] = 1$, and solve the equations so that $\tilde{Z}_i$ has the target moments.

The formulas can be stated either for marginals $\tilde{Z}_i$, or as a vector equations for all the distribution $\tilde{Z}$ at once. The only difference is the presence/absence of index i at all elements. To make the formulas easier to read, we use the vector form:

$$
\begin{aligned}
\mathbb{E}\left[\tilde{Z}\right] &= \mathbb{E}\left[\alpha\tilde{Y} + \beta\right] = \alpha\mathbb{E}\left[\tilde{Y}\right] + \beta = \beta = TARMOM_1 \quad \longrightarrow \text{ Note : } \tilde{Z} - \mathbb{E}\left[\tilde{Z}\right] = \alpha\tilde{Y} \\
Var[\tilde{Z}] &= \mathbb{E}\left[\left(\tilde{Z} - E[\tilde{Z}]\right)^2\right] = \mathbb{E}\left[(\alpha\tilde{Y})^2\right] = \alpha^2\mathbb{E}\left[\tilde{Y}^2\right] = \alpha^2 Var[\tilde{Y}] = \alpha^2 = TARMOM_2 \\
Skew[\tilde{Z}] &= \mathbb{E}\left[\left(\tilde{Z} - E[\tilde{Z}]\right)^3\right] = \ldots = \alpha^3 Skew[\tilde{Y}] = \alpha^3 MOM_3 = TARMOM_3 \\
Kurt[\tilde{Z}] &= \mathbb{E}\left[\left(\tilde{Z} - E[\tilde{Z}]\right)^4\right] = \ldots = \alpha^4 Kurt[\tilde{Y}] = \alpha^4 MOM_4 = TARMOM_4
\end{aligned}
$$

Solving for the unknown parameters we get:

$$
\begin{aligned}
\alpha &= \sqrt{TARMOM_2} \\
\beta &= TARMOM_1 \\
MOM_3 &= \frac{TARMOM_3}{\alpha^3} = \frac{TARMOM_3}{TARMOM_2^{3/2}} \\
MOM_4 &= \frac{TARMOM_4}{\alpha^4} = \frac{TARMOM_4}{TARMOM_2^2}
\end{aligned}
$$

B Matrix transformation $\tilde{Y} = L \times \tilde{X}$

We seek an n-dimensional random variable $\tilde{Y}$ with zero means, variances equal to 1, skewness MOM_3, kurtosis MOM_4 and a correlation matrix $R = L \times L^T$, where L is the lower-triangular matrix. To generate the outcomes $\mathbb{Y}$ of the random variable $\tilde{Y}$ we first generate a matrix $\mathbb{X}$ of outcomes of the random variable $\tilde{X}$ with $\tilde{X}_i$ as independent random variables, and afterwards compute $\mathbb{Y}$ as $\mathbb{Y} = L \times \mathbb{X}$. Note that the k'th moment of $\tilde{Y}$ is equal to $\mathbb{E}\left[\tilde{Y}^k\right]$ because of the zero means.

During the algorithm we use the matrix transformation in two different cases. First it is used on distributions, which are abstract objects, so we work only with their (statistical) properties. In the second part of the algorithm we transform the outcomes, which are matrices of numbers. The formulas are the same in both cases, the only difference is the use of $\tilde{X}$ in the first case and $\mathbb{X}$ in the latter.

Note that we can use both notations without changing the rest of the equations: $\tilde{X}$ is a column vector of marginals $\tilde{X}_i$, while $\mathbb{X}$ can be seen as a column 'vector' of the outcomes of the marginals $\mathbb{X}_i$. The only difference is that the $\mathbb{X}_i$ is a row vector of size s, while the $\tilde{X}_i$ is a 1-dimensional random variable.

We use the distribution-notation in the rest of the appendix. The matrix transformation in a marginal-wise (i.e. column-wise) form then is:

$$
\tilde{Y}_i = \left(L \times \tilde{X}\right)_i = \sum_{j=1}^{n} L_{ij}\,\tilde{X}_j = \sum_{j=1}^{i} L_{ij}\,\tilde{X}_j
$$

where the last equality comes from the fact that L is a lower-triangular matrix and therefore $L_{ij} = 0$ for $j > i$.

Theorem – properties of $\tilde{Y} = L \times \tilde{X}$

Assume we have a n-dimensional random variable $\tilde{X}$ with properties:

i. $\mathbb{E}\big[\tilde{X}^k\big]$ exists for $k = 1 \dots 4$

ii. $\mathbb{E}\big[\tilde{X}\big] = 0 \quad$ and $\quad \mathbb{E}\big[\tilde{X}^2\big] = 1$

iii. marginals $\tilde{X}_i$, $\tilde{X}_j$ are independent for $i \neq j$

Assume further that L is a lower-triangular matrix of size $n \times n$ such that $R = L \times L^T$, where R is a correlation matrix, i.e. R is a symmetric positive semi-definite matrix with 1's on the diagonal.

If we then define a random variable $\tilde{Y}$ as $\tilde{Y} = L \times \tilde{X}$, it has the following properties:

iv. $\mathbb{E}\big[\tilde{X}^k\big]$ exists for $k = 1 \dots 4$

v. $\mathbb{E}\big[\tilde{X}\big] = 0 \quad$ and $\quad \mathbb{E}\big[\tilde{X}^2\big] = 1$

vi. $\tilde{Y}$ has a correlation matrix $R = L \times L^T$

vii. $$\mathbb{E}\left[\tilde{Y}_i^3\right] = \sum_{j=1}^{i} L_{ij}^3 \, \mathbb{E}\left[\tilde{X}_j^3\right]$$

viii. $$\mathbb{E}\left[\tilde{Y}_i^4\right] - 3 = \sum_{j=1}^{i} L_{ij}^4 \left(\mathbb{E}\left[\tilde{X}_j^4\right] - 3\right)$$

Consequence

Under the assumptions of the theorem, with an additional assumption that R is regular (and therefore positive-definite), we can express the moments of $\tilde{X}$ as:

ix. $$\mathbb{E}\left[\tilde{X}_i^3\right] = \frac{1}{L_{ii}^3}\left(\mathbb{E}\left[\tilde{Y}_i^3\right] - \sum_{j=1}^{i-1} L_{ij}^3 \, \mathbb{E}\left[\tilde{X}_j^3\right]\right)$$

x. $$\mathbb{E}\left[\tilde{X}_i^4\right] - 3 = \frac{1}{L_{ii}^4}\left[\mathbb{E}\left[\tilde{Y}_i^4\right] - 3 - \sum_{j=1}^{i-1} L_{ij}^4 \left(\mathbb{E}\left[\tilde{X}_j^4\right] - 3\right)\right]$$

Proof

iv.
Will be proved by proving the rest.

v. & vi.
From assumptions *ii.* and *iii.* we get the variance-covariance matrix of $\tilde{X}$ as $\mathbb{E}\big[\tilde{X} \times \tilde{X}^T\big] = I$. The mean of $\tilde{Y}$ follows directly:

$$\mathbb{E}\big[\tilde{Y}\big] = \mathbb{E}\big[L \times \tilde{X}\big] = L \times \mathbb{E}\big[\tilde{X}\big] = 0$$

For proving the second part of *v.* and *vi.* we compute the variance-covariance matrix of $\tilde{Y}$:

$$\mathbb{E}\big[\tilde{Y} \times \tilde{Y}^T\big] = \mathbb{E}\big[L \times \tilde{X} \times \tilde{X}^T \times L^T\big] = L \times \mathbb{E}\big[\tilde{X} \times \tilde{X}^T\big] \times L^T = L \times I \times L^T = L \times L^T = R$$

Since R is a correlation matrix, it has 1's on a diagonal, which proves *v.* It also means that R is both the variance-covariance matrix and the correlation matrix, which proves *vi.*

vii. & ix.

$$\begin{aligned}\mathbb{E}\left[\tilde{Y}_i^3\right] &= \mathbb{E}\left[\left(\sum_{j=1}^{n} L_{ij}\,\tilde{X}_j\right)\left(\sum_{k=1}^{i} L_{ik}\,\tilde{X}_k\right)\left(\sum_{l=1}^{i} L_{im}\,\tilde{X}_l\right)\right] \\ &= \mathbb{E}\left[\sum_{j,k,l=1}^{i} L_{ij}\,L_{ik}\,L_{il}\;\tilde{X}_j\,\tilde{X}_k\,\tilde{X}_l\right] = \sum_{j,k,l=1}^{i} L_{ij}\,L_{ik}\,L_{im}\,\mathbb{E}\left[\tilde{X}_j\,\tilde{X}_k\,\tilde{X}_l\right]\end{aligned}$$

The assumption of independence of $\tilde{X}$ implies that $\mathbb{E}\big[\tilde{X}_k\tilde{X}_l\tilde{X}_m\big] = 0$ if at least one index differs from the other two. To show it, let us assume that $j \neq k$ and $j \neq l$. This means that $\tilde{X}_j$ is independent on both $\tilde{X}_k$ and $\tilde{X}_l$ and we can thus write

$$\mathbb{E}\left[\tilde{X}_j\,\tilde{X}_k\,\tilde{X}_l\right] = \mathbb{E}\left[\tilde{X}_j\right]\mathbb{E}\left[\tilde{X}_k\,\tilde{X}_m\right] = 0$$

Since two equal indices means that one is different, the only case where $\mathbb{E}\big[\tilde{X}_j\,\tilde{X}_k\,\tilde{X}_l\big] \neq 0$ is for $j = k = l$. This simplifies the triple sum to a single one and proves *vii.*:

$$\mathbb{E}\left[\tilde{Y}_i^3\right] = \sum_{j=1}^{i} L_{ij}^3\,\mathbb{E}\left[\tilde{X}_j^3\right]$$

The proof of the consequence *ix.* is straightforward since the transformation is triangular and we can write the inversion directly. We start with the $\mathbb{E}\big[\tilde{X}_1^3\big]$, and continue 'downward' to $\mathbb{E}\big[\tilde{X}_n^3\big]$:

$$\begin{aligned}\mathbb{E}\left[\tilde{X}_1^3\right] &= \frac{1}{L_{11}^3}\,\mathbb{E}\left[\tilde{Y}_1^3\right] \\ \mathbb{E}\left[\tilde{X}_i^3\right] &= \frac{1}{L_{ii}^3}\left(\mathbb{E}\left[\tilde{Y}_i^3\right] - \sum_{j=1}^{i-1} L_{ij}^3\,\mathbb{E}\left[\tilde{X}_j^3\right]\right)\end{aligned}$$

Note that we can divide by L_{ii} because we have assumed $R = L \times L^T$ to be positive-definite, so we have $L_{ii} > 0\ \forall\, i$.

viii. & x.

$$\begin{aligned}\mathbb{E}\left[\tilde{Y}_i^4\right] &= \mathbb{E}\left[\left(\sum_{j=1}^{i} L_{ij}\,\tilde{X}_j\right)\left(\sum_{k=1}^{i} L_{ik}\,\tilde{X}_k\right)\left(\sum_{l=1}^{i} L_{il}\,\tilde{X}_l\right)\left(\sum_{m=1}^{i} L_{im}\,\tilde{X}_m\right)\right] \\ &= \mathbb{E}\left[\sum_{j,k,l,m=1}^{i} L_{ij}\,L_{ik}\,L_{il}\,L_{im}\;\tilde{X}_j\,\tilde{X}_k\,\tilde{X}_l\,\tilde{X}_m\right] \\ &= \sum_{j,k,l,m=1}^{i} L_{ij}\,L_{ik}\,L_{il}\,L_{im}\,\mathbb{E}\left[\tilde{X}_j\,\tilde{X}_k\,\tilde{X}_l\,\tilde{X}_m\right]\end{aligned}$$

With the same argument as for the third moment, the assumption of independent $\tilde{X}$ again means that $\mathbb{E}\big[\tilde{X}_k\tilde{X}_l\tilde{X}_m\tilde{X}_n\big] = 0$ if at least one of the indices j, k, l, and m is different from all the others. This means that at least two indices must be the same in order to make the mean non-zero. Since three equal indices imply one different, we are left with two possibilities for non-zeroes: either all the indices are equal, or there are two different indices, each one twice.

In the former case we have $\mathbb{E}\big[\tilde{X}_j\,\tilde{X}_k\,\tilde{X}_l\,\tilde{X}_m\big] = \mathbb{E}\big[\tilde{X}_j^4\big]$. In the latter case, with for example $j = l$ and $k = m$, we get

$$\mathbb{E}\left[\tilde{X}_j\,\tilde{X}_k\,\tilde{X}_l\,\tilde{X}_m\right] = \mathbb{E}\left[\tilde{X}_j^2\,\tilde{X}_k^2\right] = \mathbb{E}\left[\tilde{X}_j^2\right]\mathbb{E}\left[\tilde{X}_k^2\right] = 1 \cdot 1 = 1$$

Since we have a sum over all the combinations of indices k, l, m and n, the latter case occurs $\binom{4}{2} = 6$ times for every pair of indices. We thus have to multiply that element by 6 and sum it over $k < l$ to prevent the element from being counted twice.

We have thus simplified the sum to:

$$\mathbb{E}\left[\tilde{Y}_i^4\right] = \sum_{j=1}^{i} L_{ij}^4\, \mathbb{E}\left[\tilde{X}_j^4\right] + 6 \sum_{\substack{j,k=1 \\ j<k}}^{i} L_{ij}^2\, L_{ik}^2$$

Now we need to use the fact that L is a matrix from the Cholesky decomposition of the correlations matrix R. Since R has 1's on a diagonal the following relationship holds:

$$1 = R_{ii} = \left(L \times L^T\right)_{ii} = \sum_{j=1}^{n} L_{ij}\left(L^T\right)_{ji} = \sum_{j=1}^{n} L_{ij}^2 = \sum_{j=1}^{i} L_{ij}^2 = 1$$

Using this result, we can eliminate the double-sum from the formula for $\mathbb{E}\left[\tilde{Y}_i^4\right]$. To do this, we express the double-sum in two different ways and then sum them together.
Note that the first equality in the second line comes from a simple switching of letters.

$$\sum_{\substack{j,k=1 \\ j<k}}^{i} L_{ij}^2 L_{ik}^2 = \begin{cases} \displaystyle\sum_{j=1}^{i}\sum_{k=j+1}^{i} L_{ij}^2\, L_{ik}^2 = \sum_{j=1}^{i} L_{ij}^2 \sum_{k=j+1}^{i} L_{ik}^2 \\ \displaystyle\sum_{k=1}^{i}\sum_{j=1}^{k-1} L_{ij}^2\, L_{ik}^2 = \sum_{j=1}^{i}\sum_{k=1}^{j-1} L_{ik}^2\, L_{ij}^2 = \sum_{j=1}^{i} L_{ij}^2 \sum_{k=1}^{j-1} L_{ik}^2 \end{cases}$$

$$\begin{aligned} 2\sum_{\substack{j,k=1 \\ j<k}}^{i} L_{ij}^2 L_{ik}^2 &= \sum_{j=1}^{i} L_{ij}^2 \left(\sum_{k=1}^{j-1} L_{ik}^2 + \sum_{k=j+1}^{i} L_{ik}^2\right) = \sum_{j=1}^{i} L_{ij}^2 \left(\sum_{k=1}^{i} L_{ik}^2 - L_{ij}^2\right) \\ &= \left(\sum_{j=1}^{i} L_{ij}^2\right)\left(\sum_{k=1}^{i} L_{ik}^2\right) - \sum_{j=1}^{i} L_{ij}^4 = 1 - \sum_{j=1}^{i} L_{ij}^4 \end{aligned}$$

Now we substitute the result into the formula for $\mathbb{E}\left[\tilde{Y}_i^4\right]$ and get:

$$\begin{aligned} \mathbb{E}\left[\tilde{Y}_i^4\right] &= \sum_{j=1}^{i} L_{ij}^4\, \mathbb{E}\left[\tilde{X}_j^4\right] + 6 \sum_{\substack{j,k=1 \\ j<k}}^{i} L_{ij}^2\, L_{ik}^2 = \sum_{j=1}^{i} L_{ij}^4\, \mathbb{E}\left[\tilde{X}_j^4\right] + 3 - 3\sum_{j=1}^{i} L_{ij}^4 \\ &= \sum_{j=1}^{i} L_{ij}^4 \left(\mathbb{E}\left[\tilde{X}_j^4\right] - 3\right) + 3 \end{aligned}$$

To prove *viii.* we need only subtract 3 from both sides of the formula. Note that we get a formula with kurtosis decreased by 3, i.e. with kurtosis relative to $\mathcal{N}(0,1)$:

$$\mathbb{E}\left[\tilde{Y}_i^4\right] - 3 = \sum_{j=1}^{i} L_{ij}^4 \left(\mathbb{E}\left[\tilde{X}_j^4\right] - 3\right)$$

Since the formula has the same form as the one for kurtosis, the second part of the consequence is obtained the same way as for kurtosis. This concludes the proof.

$$\begin{aligned} \mathbb{E}\left[\tilde{X}_1^4\right] - 3 &= \frac{1}{L_{11}^4}\, \mathbb{E}\left[\tilde{Y}_1^4\right] - 3 \\ \mathbb{E}\left[\tilde{X}_i^4\right] - 3 &= \frac{1}{L_{ii}^4}\left[\mathbb{E}\left[\tilde{Y}_i^4\right] - 3 - \sum_{j=1}^{i-1} L_{ij}^4 \left(\mathbb{E}\left[\tilde{X}_j^4\right] - 3\right)\right] \end{aligned}$$

Q.E.D.

Remarks

- Note that the set of moments $(0, 1, 0, 3)$ is an invariant of this transformation. This just confirms the known theoretical result that the linear transformations preserve normality. In the context of our algorithm it means that if we generate normal variables, we can skip Step 3 of the algorithm.
- Even if we know that the regularity of R gives us $L_{ii} \neq 0$, we still can have $L_{ii} \approx 0$ in a case of 'almost-singular' correlation matrix R. As we divide by L_{ii}^3 and L_{ii}^4 during the inverse transformation, the chance of numerical instability is substantial. The result of those instabilities can be either extreme values of the target skewness or kurtosis of $\tilde{X}$, or even a negative kurtosis.

  The important question is thus when this problem occurs, i.e. when has $\tilde{Y}$ an almost-singular correlation matrix R? The basic answer is that this happens if one $\tilde{Y}_i$ is almost a linear combination of the others, in other words, it can be almost fully explained using the other $\tilde{Y}_j$. The most obvious example is a correlation matrix with very high numbers (abs. value > 0.9), yet we should be aware that with increasing n (dimension of $\tilde{Y}$) we increase the chance of having a dependency also with 'moderate' correlations.

  To avoid the possible problems, we should do some tests before starting the algorithm. The easiest (but not sufficient) is to check if $L_{ii} \geq \varepsilon > 0$ for all asset classes i. Note that if we use the method from *Lurie and Goldberg, 1998* to ensure positive semi-definiteness of the correlation matrix, we can simply add the constraints $\{L_{ii} \geq \varepsilon > 0\}$ to the QP model.

  Although more difficult, it would be advisable to check the condition number of the correlation matrix. This number is defined as the ratio $\frac{\lambda_{max}(R)}{\lambda_{min}(R)}$ of maximum and minimum eigenvalues of the matrix R and measures how ill-conditioned the matrix is. Higher values indicate that the matrix is close to singular, so the inverse transformation can return incorrect results.

  In case the above precautions fail, we suggest a simple heuristic for handling the singularity and 'almost-singularity' problem inside our algorithm – see Section 2.5.1. It is a crude procedure, but it should be sufficient in most cases.

C Cubic transformation $\tilde{Y} = a + b\tilde{X} + c\tilde{X}^2 + d\tilde{X}^3$

The purpose of this transformation is to generate a univariate random variable $\tilde{Y}_i$ with specified first four moments $\mathbb{E}\big[\tilde{Y}_i^k\big], k = 1..4$, given the random variable $\tilde{X}_i$ with known first 12 moments $\mathbb{E}\big[\tilde{X}_i^k\big], k = 1..12$.

The problem is to find the transform parameters a_i, b_i, c_i and d_i. For this we have to express $\mathbb{E}\big[\tilde{Y}_i^k\big]$ as functions of $\mathbb{E}\big[\tilde{X}_i^k\big]$. The formulas can be stated either for marginals $\tilde{Y}_i$, or as a vector equations for all the distribution $\tilde{Y}$ at once. The only difference is the presence/absence of index i at all elements. To make the formulas easier to read, we use the vector form.

$$
\begin{aligned}
\mathbb{E}\big[\tilde{Y}\big] &= a + b\mathbb{E}\big[\tilde{X}\big] + c\mathbb{E}\big[\tilde{X}^2\big] + d\mathbb{E}\big[\tilde{X}^3\big] \\
\mathbb{E}\big[\tilde{Y}^2\big] &= d^2\mathbb{E}\big[\tilde{X}^6\big] + 2cd\mathbb{E}\big[\tilde{X}^5\big] + (2bd + c^2)\mathbb{E}\big[\tilde{X}^4\big] + (2ad + 2bc)\mathbb{E}\big[\tilde{X}^3\big] \\
&+ (2ac + b^2)\mathbb{E}\big[\tilde{X}^2\big] + 2ab\mathbb{E}\big[\tilde{X}\big] + a^2 \\
\mathbb{E}\big[\tilde{Y}^3\big] &= d^3\mathbb{E}\big[\tilde{X}^9\big] + 3cd^2\mathbb{E}\big[\tilde{X}^8\big] + (3bd^2 + 3c^2d)\mathbb{E}\big[\tilde{X}^7\big] + (3ad^2 + 6bcd + c^3)\mathbb{E}\big[\tilde{X}^6\big] \\
&+ (6acd + 3b^2d + 3bc^2)\mathbb{E}\big[\tilde{X}^5\big] + (a(6bd + 3c^2) + 3b^2c)\mathbb{E}\big[\tilde{X}^4\big] \\
&+ (3a^2d + 6abc + b^3)\mathbb{E}\big[\tilde{X}^3\big] + (3a^2c + 3ab^2)\mathbb{E}\big[\tilde{X}^2\big] + 3a^2b\mathbb{E}\big[\tilde{X}\big] + a^3 \\
\mathbb{E}\big[\tilde{Y}^4\big] &= d^4\mathbb{E}\big[\tilde{X}^{12}\big] + 4cd^3\mathbb{E}\big[\tilde{X}^{11}\big] + (4bd^3 + 6c^2d^2)\mathbb{E}\big[\tilde{X}^{10}\big] \\
&+ (4ad^3 + 12bcd^2 + 4c^3d)\mathbb{E}\big[\tilde{X}^9\big] + (12acd^2 + 6b^2d2 + 12bc^2d + c^4)\mathbb{E}\big[\tilde{X}^8\big] \\
&+ (a(12bd^2 + 12c^2d) + 12b^2cd + 4bc^3)\mathbb{E}\big[\tilde{X}^7\big] + (6a^2d^2 + a(24bcd + 4c^3) \\
&+ 4b^3d + 6b^2c^2)\mathbb{E}\big[\tilde{X}^6\big] + (12a^2cd + a(12b^2d + 12bc^2) + 4b^3c)\mathbb{E}\big[\tilde{X}^5\big] \\
&+ (a2(12bd + 6c^2) + 12ab^2c + b^4)\mathbb{E}\big[\tilde{X}^4\big] + (4a^3d + 12a^2bc + 4ab^3)\mathbb{E}\big[\tilde{X}^3\big] \\
&+ (4a^3c + 6a^2b^2)\mathbb{E}\big[\tilde{X}^2\big] + 4a^3b\mathbb{E}\big[\tilde{X}\big] + a^4
\end{aligned}
$$

Note that since the matrix transformation $\tilde{Y}_i = L \times \tilde{X}_i$ does not change the first two moments, we have $E[\tilde{X}_i] = 0$ and $E[\tilde{X}_i^2] = 1$. We could thus simplify the above formulas slightly by assuming that the first two moments are exactly 0 and 1. In our implementation we have, however, used the formulas in the presented form, computing the actual moments.

There are several possible ways to solve this system for the coefficients a, b, c and d. The most efficient is probably to write a special code (for example in `C++`) that will use some numerical method (like Newton's) to solve the system.

We have, however, used a different approach and formulated a non-linear regression with a, b, c and d as independent variables and moments of $\tilde{Y}_i$ as dependent variables, and used a non-linear solver to minimise the distance of actual moments from their target values.

Another possibility, is to use a *goal seek* function that exists for example in `Excel` or `Matlab`.